Statistical Techniques in Robotics (16-831, S21) Lecture #16 (Wednesday, March 31)

Model-based Value-based RL

Lecturer: Kris Kitani *Scribes: Chih-Wei Wu, Yu-Jhe Li*

1 Review

In the previous course, we have introduced the general sequence feedback learning problem and we will review these concepts in this section. Generally, most of the contents in this document are from course materials [4].

1.1 Sequence Feedback Learning Problem

Before we move into the RL problems that are sequential, evaluative, and sampled, we first review some of the learning problem we learned in the Table 1. Difference of these problems are summarized as follows:

- **Instructive** problems (PWMA,OLC): Observe the fully observable loss, update all your predictor parameters.
- **Sampled-evaluative** problems (MAB, C-MAB): observe the partially observable loss, update one (arm) predictor parameter at a time.
- **Sequential** problems (RL): Obtain a sequence of rewards, update your future predictor (value function), then update action predictor (policy).

Table 1: Decision-making problems covered so far

Problem	Sampled	Evaluative	Sequential
PWMA	✗	✗	✗
OLC	✓	✗	✗
MAB	✗	✓	✗
C-MAB	✓	✓	✗
RL	✓	✓	✓

1.2 Sequential Decision Making and Markov Decision Process

We have different ways to decompose a temporal sequence of variables such as:

- Bayesian Bandit: $p(r_0, ..., r_T|a_0, ..., a_T, \theta) = \prod_{t=0}^{T} p(r_t|a_t, \theta)$
- Dynamic Bayesian Network: $p(x_0, ..., x_T, e_1, ..., e_T) = p(x_0) \prod_{t=0}^{T} p(x_{t+1}|x_t)p(e_t|x_t)$

Today, we introduce the **Markov Decision Process**:

$$p(s_0, ..., s_T, a_0, ..., a_T) = p_0(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)p(a_t|s_t)$$

A Markov Decision Process (MDP) can be used as a framework to formulate and solve reinforcement learning problems. In MDP, we define some of the important notations as: state $s \in S$, action $a \in A$, state transition dynamics $p(s'|s,a)$, reward function $r(s',s,a)$, state prior $p_0(s)$, policy $\pi(a|x)$, and discount factor γ. Policy describes which action to take in a given state. State transition dynamic describes the probability of transition to another state. Reward function maps a state to a real value. The graphical model is presented in the Figure 1.

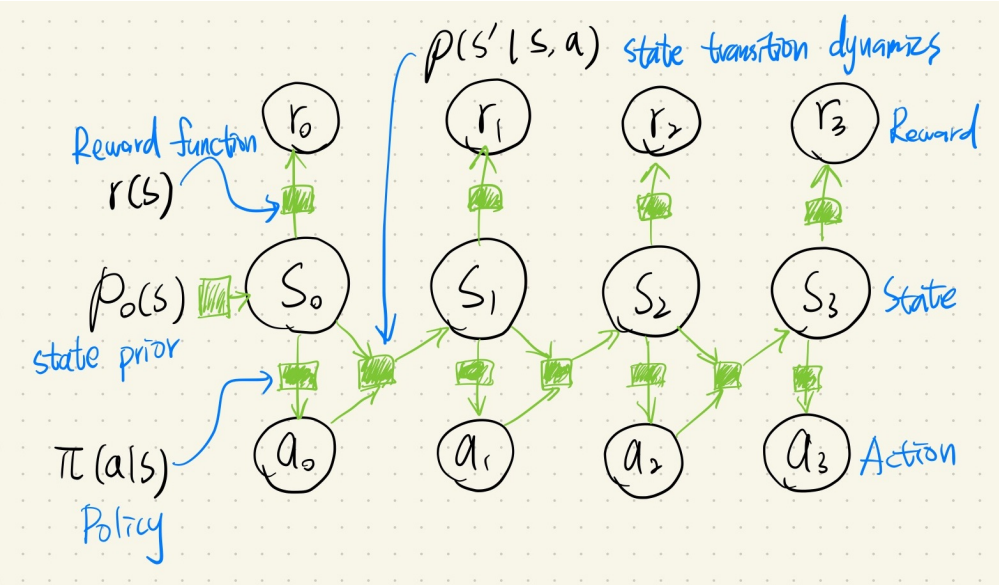

Figure 1: Graphical model of MDP.

1.3 Value function in MDP

We define the total expected return of a trajectory starting in state s as **state value function**:

$$V^{\pi}(s) = \mathbb{E}_p[r_0 + r_1 + r_2 + ...|s_0 = s],$$

where the reward is defined as:

$$r_t \triangleq r(s_{t+1} = s', a_t = a, s_t = s).$$

By a discount factor $\gamma = [0, 1]$, we can also define the infinite horizon discounted return as:

$$V^{\pi}(s) = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r^1 + \gamma^2 r^2 + ...|s_0 = s].$$

The total expected return of a trajectory starting in state s and taking action a is defined as **state-action value function**:

$$Q^{\pi}(s,a) = \mathbb{E}_p[\gamma^0 r(s_0) + \gamma^1 r(s_1) + \gamma^2 r(s_2) + ...|s_0 = s, a_0 = a].$$

The relationship between state value function (V) and state-action value function (Q) is:

$$V^{\pi}(s) = \sum_a \pi(a|s) Q^{\pi}(s,a)$$

2 Summary

2.1 MDP Concepts

We have reviewed the value functions in the section 1.3. Now we are going to introduce more concepts in the MDP which include the Bellman equation and the Bellman Optimality Equations.

2.1.1 The Bellman equation

Now we would like to make the value function to be recurrent. That is, the recursive relationship between state value functions under a given policy is defined as Bellman equation:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} p(s'|s,a)[r(s',a,s) + \gamma V^{\pi}(s')]$$

This can be interpreted as taking one step, receiving a reward and see the value of the state where you transition to. The current state value will consider all the possible transitions to neighboring state. *Proof:*

$$\begin{aligned}
V^{\pi}(s_0) =& \mathbb{E}\left[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + ... \middle| s_0\right] \\
=& \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0\right] \\
=& \mathbb{E}\left[r_0 + \sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0\right] \\
=& \sum_{a_0:\infty} \sum_{s_1:\infty} p(s_{1:\infty}, a_{0:\infty}) \left[r_0 + \sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0\right] \\
=& \sum_{a_0} \sum_{s_1} p(s_1|s_0,a_0)\pi(a_0|s_0) \left\{ r_0 + \sum_{a_1:\infty} \sum_{s_2:\infty} p(s_{2:\infty, a_{1:\infty}}) \left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1\right]\right\} \\
=& \sum_{a_0} \sum_{s_1} p(s_1|s_0,a_0)\pi(a_0|s_0) \left\{ r_0 + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1\right]\right\} \\
=& \sum_{a_0} \pi(a_0|s_0) \sum_{s_1} p(s_1|s_0,a_0) \left\{r_0 + \gamma V^{\pi}(s_1)\right\}
\end{aligned}$$

The Bellman Equation for state-action value function can also be defined as:

$$Q^{\pi}(s,a) = \sum_{s'} p(s'|s,a) \left\{ r(s',a,s) + \sum_{a'} \pi(a'|s') Q^{\pi}(s',a') \right\}$$

Proof:

$$\begin{aligned}
Q^{\pi}(s_0, a_0) =& \mathbb{E}\left[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + ... \middle| s_0, a_0\right] \\
=& \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0, a_0\right] \\
=& \mathbb{E}\left[r_0 + \sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0, a_0\right] \\
=& \sum_{s_1:\infty} p(s_{1:\infty}, a_{0:\infty}) \left[r_0 + \sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0, a_0\right] \\
=& \sum_{s_1} p(s_1|s_0, a_0) \left\{r_0 + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1\right]\right\} \\
=& \sum_{s_1} p(s_1|s_0, a_0) \left\{r_0 + \sum_{a_1} \pi(a_1|s_1) \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1, a_1\right]\right\} \\
=& \sum_{s_1} p(s_1|s_0, a_0) \left\{r_0 + \sum_{a_1} \pi(a_1|s_1) Q^{\pi}(s_1, a_1)\right\}
\end{aligned}$$

The importance of Bellman equation:

- It defines a (recursive) relationship between value functions
- It is used to derive the Bellman optimality equation
- It gives us a useful constraint for optimization
- It provides the basis for many algorithms to solve for the optimal value function and optimal policy.

2.1.2 The Bellman Optimality Equations

We can now use these equations to derive the Bellman optimality equations. This gives us a method of comparing policies and can be used to find the best policy. In this case, the policy that results in the agent collecting the maximum reward is regarded as the best policy and it should obtain the greatest expected return.

$$V^{\pi^*}(s) = \max_{\pi} V^{\pi^*}(s)$$

$$Q^{\pi^*}(s, a) = \max_{\pi} Q^{\pi^*}(s, a)$$

Expanding the equations we can get:

$$V^{\pi^*}(s) = \max_{a} \sum_{s'} p(s'|s, a)[r_t + \gamma V^{\pi^*}(s')]$$

$$Q^{\pi^*}(s, a) = \sum_{s'} p(s'|s, a)[r(s) + \gamma \max_{a'} Q^{\pi^*}(s', a')]$$

We can further derive the relationship between Q and V as:

- From Q to V:
$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a)$$
- From V to Q:
$$Q^{\pi^*}(s,a) = \sum_{s'} p(s'|s,a)[r(s) + \gamma V^{\pi^*}(s')]$$

Proof:

$$\begin{aligned}
V^{\pi^*}(s) &= \sum_a \pi(a|s) Q^{\pi^*}(s,a) \\
&= \max_a Q^{\pi^*}(s,a) \\
&= \max_a \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right] \\
&= \max_a \mathbb{E}\left[r_0 + \sum_{t=0}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right] \\
&= \max_a \sum_{s'} p(s_1 = s'|s,a) \left[r_0 + \mathbb{E}\left\{\sum_{t=0}^{\infty} \gamma^t r_t \middle| s_1 = s'\right\}\right] \\
&= \max_a \sum_{s'} p(s'|s,a)[r_t + \gamma V^{\pi^*}(s')]
\end{aligned}$$

2.2 Model-based Prediction and Control

Recall in Reinforcement Learning, our learning goal is to **obtain the optimal policy** $\hat{\pi}$ from a data distribution $D(\zeta, R)$. The optimal policy $\hat{\pi} : s \to a$ is a function that maps a state (or observation x) to an action. The data distribution could be sampled to generate a data sample consisting a sequence of observation and actions $\zeta = \{(s_1, a_1), (s_2, a_2), \ldots, (s_T, a_T)\}$, and a return value $R_i \in \mathbb{R}$ that tells you how good/bad the entire sequence is. The optimal policy is obtained **by optimizing for the maximum expected reward**:

$$\hat{\pi} = \operatorname*{argmax}_{\pi} \mathbb{E}_{p_\pi(\zeta)}\Big[\sum_{t=0}^{T} r^t\Big]. \tag{1}$$

Here, $r^t \triangleq r(s^{t+1}, a^t, s^t)$ is the reward of iteration t, consisting the current state s^t, the current action a^t and the next state s^{t+1}. $\zeta = \{s^{(0)}, a^{(0)}, s^{(1)}, a^{(1)}, ..., s^{(T)}, a^{(T)}\}$ is the state-action sequence. $p_\pi(\zeta) = p(s^0) \prod_{t=0}^{T} \pi(a^t|s^t) p(s^{t+1}|s^t, a^t)$ is the probability of the entire sequence.

In previous lecture, we quantify the expected total return of a trajectory starting in state s and taking action a as a **state-action value function**:

$$Q^\pi(s,a) = \mathbb{E}_p[\gamma^0 r(s_0) + \gamma^1 r(s_1) + \ldots | s_0 = s, a_0 = a]. \tag{2}$$

So the optimization goal could be rewrite as:

$$\pi(s) = \operatorname*{argmax}_a Q(s,a) \tag{3}$$

If we could solve for the value function Q, then it is easy to derive the policy. This kind of RL approach is called the **Value-based RL**. See Figure 2 for more context. For a value-based RL problem, we are given the following as input:

- Finite state space $s \in \mathcal{S}$
- Finite action space $a \in \mathcal{A}$
- Dynamic model $p(s'|s, a)$
- Reward function $r(s', s, a)$

and begin asked to predict the optimal policy. But is it all? Recall the state-action value function is actually characterized by a policy $\pi(a|s)$. However the policy is what we want to solve in the first place. So this could be though as an chicken-and-egg problem.

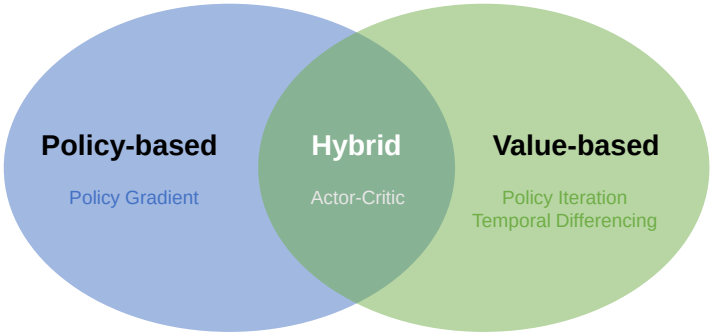

Figure 2: Value-based Reinforcement Learning.

To solve this problem, we typically fix the policy calculated from the last iteration, and solve for the value function. We call this step **Policy Evaluation**. We then retrieve the policy that maximizes the value function, and use this policy for the next iteration. We term this step as **Policy Improvement**. We alternate these two steps throughout the algorithm. An illustration of the process is presented in Figure 3.

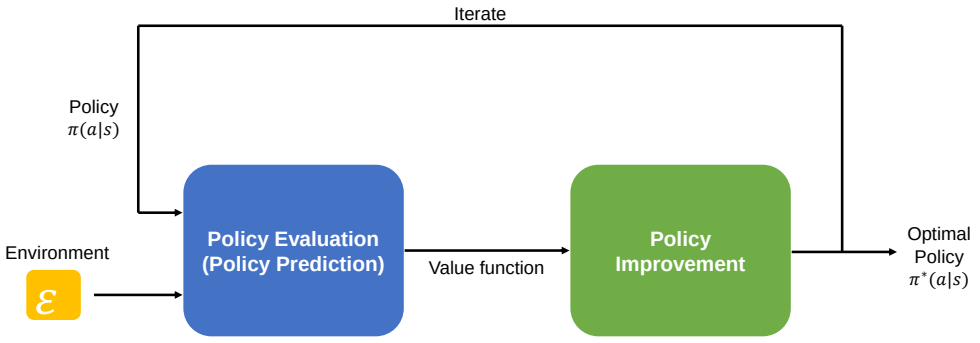

Figure 3: Solving Value-based RL.

Depending on the type of environment given to Policy Evaluation step, we can categorize value-based RL into 2 classes:

1. The **model-based** method assumes the state transition dynamics and reward function are known.
2. The **model-free** method assumes we do not know anything about the environment, giving the algorithm only sampled feedback.

In this section, we will focus on value-based model-based method (a.k.a Policy Iteration).

2.3 Policy Evaluation

First, we focus on the Policy Evaluation step for solving value-based model-based RL, as indicated in Figure 3. For this step, we take policy $\pi(s|a)$, transition dynamics $p(s'|s,a)$ and $r(s',s,a)$ as input, and aims to predict the value function characterized by the Bellman Equation:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s',a,s) + \gamma V^\pi(s')] \tag{4}$$

$$= \sum_a \pi(a|s) Q^\pi(s,a) \tag{5}$$

We could develop an algorithm for prediction based on this objective:

Algorithm 1 Policy Evaluation $(\pi, r(s), p(s'|s,a), \gamma)$

$V \leftarrow \text{rand}(\mathbb{R})$
$V' \leftarrow \text{rand}(\mathbb{R})$
while $\max_s |V(s) - V'(s)| \geq \epsilon$ **do**
  $V' \leftarrow V$
  for $s \in \mathcal{S}$ **do**
    $Q(s,a) = r(s) + \gamma \sum_{s'} p(s'|s,a) V'(s') \quad \forall a$ ▷ This step interact with the environment
    $V(s) = \sum_a \pi(s|a) Q(s,a)$
  end for
end while
return Q

2.4 Policy Improvement

For the Policy Improvement step, we take the value function $Q(s,a)$ given by the Policy Evaluation step, and estimate the optimal policy $\pi(a|s)$ for next iteration. Recall that when the policy is optimal, Equation 5 could be rewrite as:

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a),$$

assuming that the transition of state is deterministic to the action. Then, we write the algorithm for Policy Improvement as follows.

Algorithm 2 Policy Improvement (Q)

for $s \in \mathcal{S}$ **do**
  $\pi(s) = \text{argmax}_a Q(s,a)$
end for
return π

2.5 Policy Iteration

Now, we can put together all the pieces to form our algorithm. The algorithm for Policy Iteration is as follows.

Algorithm 3 Policy Iteration $(r(s), p(s'|s,a), \gamma)$

1: $\pi \leftarrow \text{rand}(\mathcal{A})$
2: $Q \leftarrow \textsc{PolicyEvaluation}(\pi, r(s), p(s'|s,a), \gamma)$
3: $\pi'(s) \leftarrow \textsc{PolicyImprovement}(Q)$
4: **if** $\pi' = \pi$ **then**
5: **return** π
6: **end if**
7: Go to line 2

2.6 Value Iteration

In practice, Policy Iteration takes a long time to converge. This is because we first do a double loop in Policy Evaluation until convergence on the value function V, then we would do another double loop in Policy Improvement to update the policy. To speed things up, we **Value Iteration** combine the Policy Evaluation and Policy Improvement step together.

Algorithm 4 Value Iteration $(r(s), p(s'|s,a), \gamma)$

1: $\pi \leftarrow \text{rand}(\mathcal{A})$
2: $V \leftarrow \text{rand}(\mathbb{R})$
3: $V' \leftarrow \text{rand}(\mathbb{R})$
4: **while** $\max_s |V(s) - V'(s)| \geq \epsilon$ **do**
5: $V' \leftarrow V$
6: **for** $s \in \mathcal{S}$ **do**
7: $Q(s,a) = r(s) + \gamma \sum_{s'} p(s'|s,a) V'(s') \quad \forall a$
8: $\pi'(s) = \text{argmax}_a \, Q(s,a)$ ▷ Policy Improvement is incorporated here
9: $V(s) = \sum_a \pi(s|a) Q(s,a)$
10: **end for**
11: **end while**
12: **return** π

3 Appendix

3.1 Convergence of Value Iteration

There are a couple of theorems ensuring the convergence of Value Iteration algorithm [3, 5]. The first is presented by Bertsekas et al. [1].

Theorem 1. *V converges to V^*, if each state is visited infinitely often.*

This theorem states the Value Iteration would eventually converge to the optimal value function,

which would also yield the optimal policy. The theorem also state that the assignment of V does not have to be in strict order as shown in the Value Iteration algorithm, but instead can occur asynchronously in parallel provided that each state is visited infinitely often.

Theorem 2. *If* $\max_s |V^{t+1}(s) - V^t(s)| < \epsilon$, *then* $\max_s |V^{t+1}(s) - V^*(s)| < \frac{2\epsilon\gamma}{1-\gamma} \quad \forall s.$

The second theorem presented by William and Baird et al. [**?**] states that if the maximum difference between to successive states is less that ϵ, than the value function differs from the value function of optimal policy by no more than $2\epsilon\gamma/(1-\gamma)$. This provides a reasonable stopping criteria for the value function.

Theorem 3. *Greedy policy will be optimal in a finite number of steps (even if not converged to optimal value function).*

The third theorem presented by Bertsekas et al. [2] shows that the greedy policy used by Value Iteration is guaranteed to be optimal in a finite number of steps, even though the value function may not have converged. In practice, the greedy policy often reach optimal long before the value function has converged.

3.2 Time Complexity of Policy Iteration and Value Iteration

By observing the algorithm of Policy Iteration and Value Iteration, we could see that the time complexity of Policy Iteration is larger than that of Value Iteration. For Value Iteration, each iteration takes $O(|\mathcal{A}||\mathcal{S}|^2)$ steps. In comparison, Policy Iteration takes $O(|\mathcal{A}||\mathcal{S}|^2 + |\mathcal{S}|^3)$ in each iteration. However, in practice, Policy Iteration takes fewer iterations to converge although its iteration takes longer [3, 5].

References

[1] D. Bertsekas and J. Tsitsiklis. Parallel and distributed computation: Numerical methods, prentice hall. *New Jersey: Englewood Cliffs*, 1989.

[2] F. J. Beutler. Dynamic programming: Deterministic and stochastic models (dimitri p. bertsekas). *SIAM Review*, 31(1):132, 1989.

[3] CMU. Course 10601+10301, spring 2021, lecture 15 slide.

[4] CMU. Course 16831, spring 2021, lecture 16 slide.

[5] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.